Imperial College London

Algorithmic Trading in Julia using Predictive Control

Student: Bruno Castro
Supervisors: Eric Kerrigan, Lucian Nita

AirBorne.jl

https://github.com/JuDO-dev/AirBorne.jl [2]

"A complete algorithmic trading framework in Julia."

Motivation

Julia is a relatively new highly expressive high performant programming language that due to lack of adoption has its package ecosystem underdeveloped with respect to other more widely adopted languages such as C, Java and Python.

Contribution: In particular in the context of algorithmic trading there is a vacuum in the availability of well-maintained packages capable of generic algorithmic trading and backtesting, unlike in C or Python were tools such as Zorro and Zipline are widely used and maintained by its community.

In order to address this gap this project developed an Imperial College backed software "AirBorne" with the main goal of it being the package of reference for algorithmic trading in Julia.

Predictive Control: Moreover, the research in algorithmic trading strategies is an area of increasing interest in the control community, with active research in the area of model predictive control (MPC).

This project replicated a recently published state-of-the-art Mean Variance MPC strategy using Hidden Markov Models [1] as a modelling technique and compare its performance against similar models with different models and different well stablished strategies.

Conclusion: The project successfully put forward a fully functional though restricted algorithmic trading platform, accessible for anyone to use, adding immediate value to the Julian algorithmic trading community whilst providing a relevant use case for current state of the art predictive control in algorithmic trading.

MV-MPC performance

The usage of predictive control through the trading strategy Mean Variance MPC (MV-MPC) [1] outperformed well established strategies in the trading industries such as Simple Moving Average and Markowitz.

Figure 3 provides a comparison of the accumulated return of the portfolio following different trading strategies. Equation 1 presents the optimal control problem formulation of the MV-MPC.

OCP

$$\max_{\pi_{t+1},\dots,\pi_{t+H}} \left(\sum_{\tau=t+1}^{\tau=t+H} \hat{r}_{\tau|t}^T \pi_\tau - \gamma^{risk} \left(\pi_\tau^T \hat{\Sigma}_{\tau|t} \pi_\tau \right) - \gamma^{trade} ||\pi_\tau - \pi_{\tau-1}||_1 \right) \quad (1)$$

s.t.,

$$\pi_\tau \geq 0 \quad \forall \tau = t+1, \dots, t+H$$

$$\mathbf{1}^T \pi_\tau = 1 \quad \forall \tau = t+1, \dots, t+H$$

π_τ: Portfolio distribution at time τ	H: Horizon
$\hat{r}_{\tau\|t}$: Expected return at time τ as of time t	γ^{risk}: Risk weight factor
$\hat{\Sigma}_{\tau\|t}$: Covariance matrix at time τ as of time t	γ^{trade}: Sell/buy cost

Predictive Model

The MV-MPC models the returns of each asset in a portfolio as a series of random variables which is a time varying stochastic process, at each point in time an estimate of its expected value and covariance matrix is produced for the horizon of the OCP.

Four different models were studied, Behavioural model*, HMM, Linear regression and Last value of those HMM was found to be the best performant in a dataset of 22 companies during 2020 as shown in Figure 4.

* The behavioural model tested, was an early version of the method and more sophisticated approaches are available (though not tested for this experiment).

Architecture

AirBorne is meant to be a Julia package for advanced algorithmic trading, its modular design allows for customizations on hypothesis of market models, robust handling of data and transparent benchmarking and analysis of trading strategies.

In its first iteration AirBorne counts with its data layer and back testing framework, but its design allows the placement of a Live Trading feature to seamlessly transfer strategies from idealized simulations to the real thing. Figures 1 and 2 display the internal distribution of features in the package and the flow of data when using it.

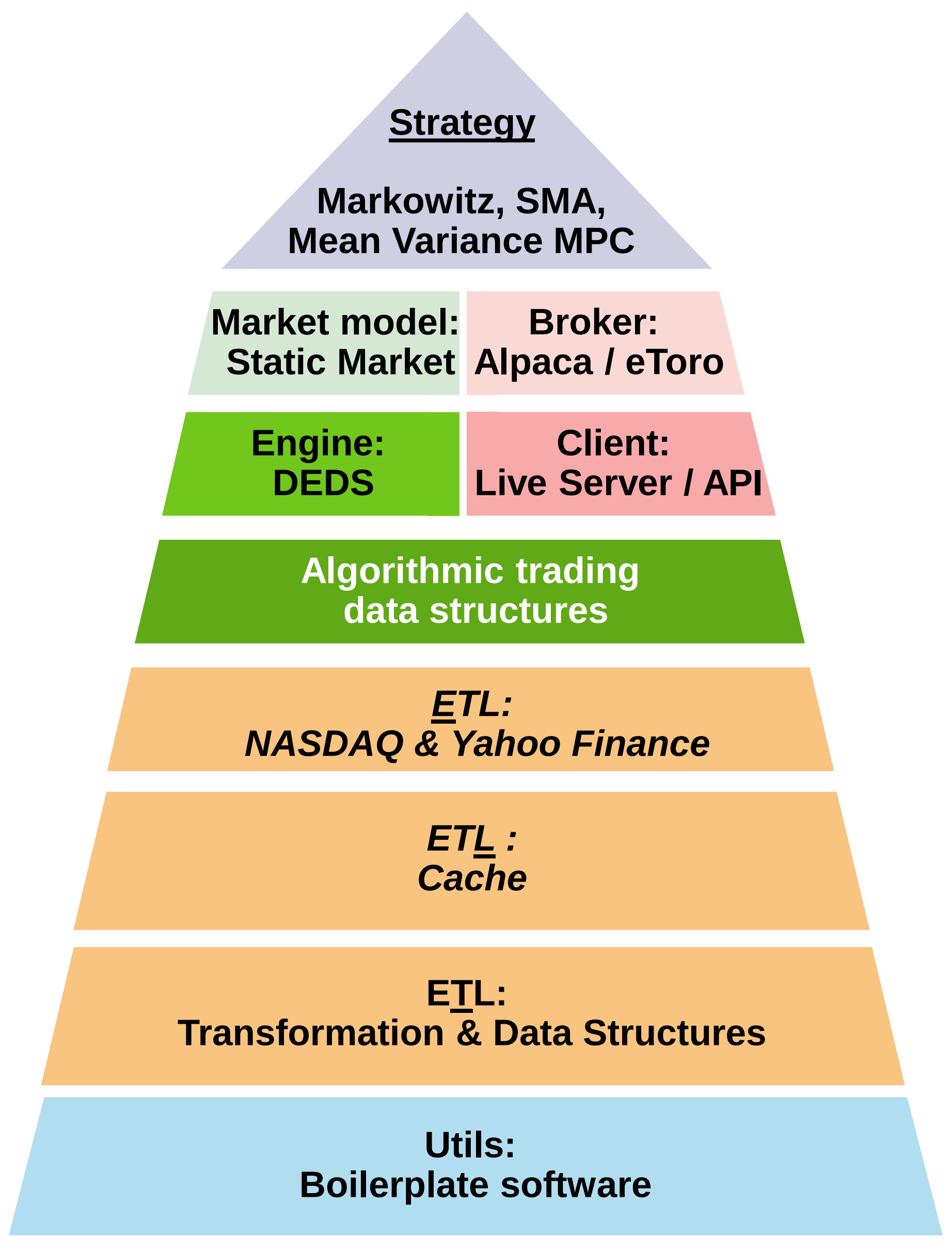

Strategy
Markowitz, SMA, Mean Variance MPC
Market model: Static Market
Broker: Alpaca / eToro
Engine: DEDS
Client: Live Server / API
Algorithmic trading data structures
ETL: NASDAQ & Yahoo Finance
ETL : Cache
ETL: Transformation & Data Structures
Utils: Boilerplate software

Figure 1: Composition and module dependencies within AirBorne.jl

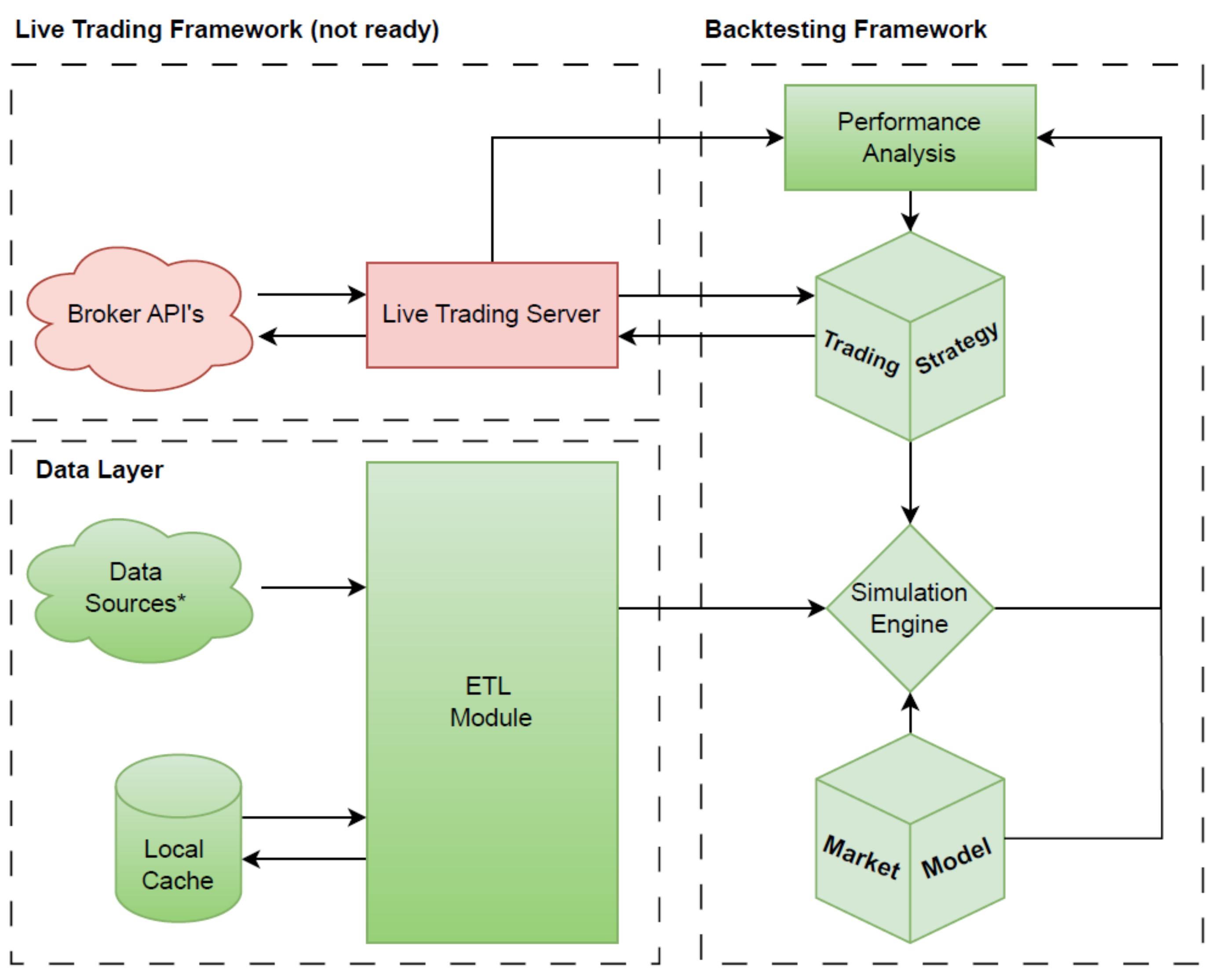


Figure 2: Architecture and flow of data withing in AirBorne.jl

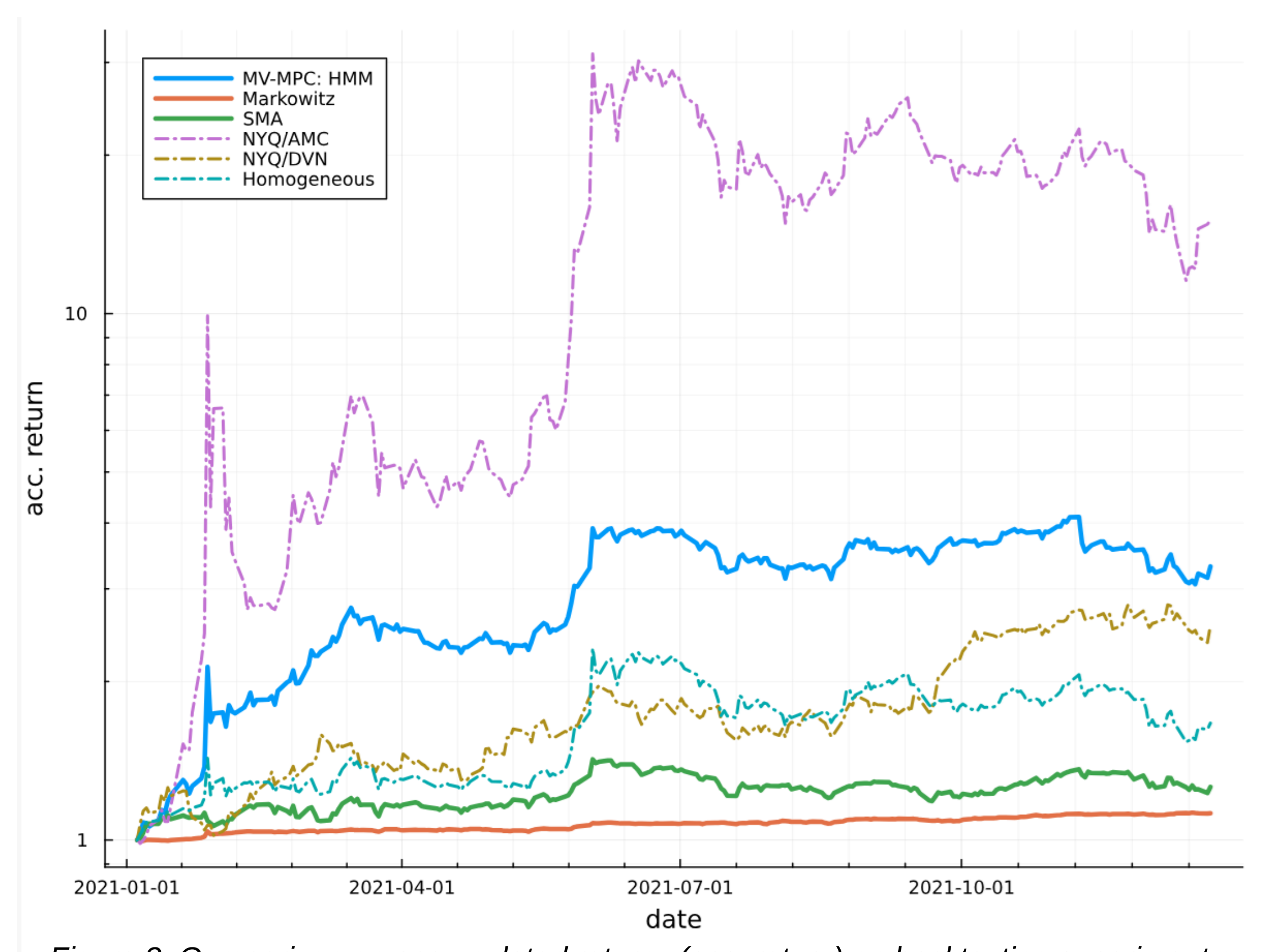


Figure 3: Comparison on accumulated returns (acc. return) on backtesting experiment on 22 liquid US stocks during 2021

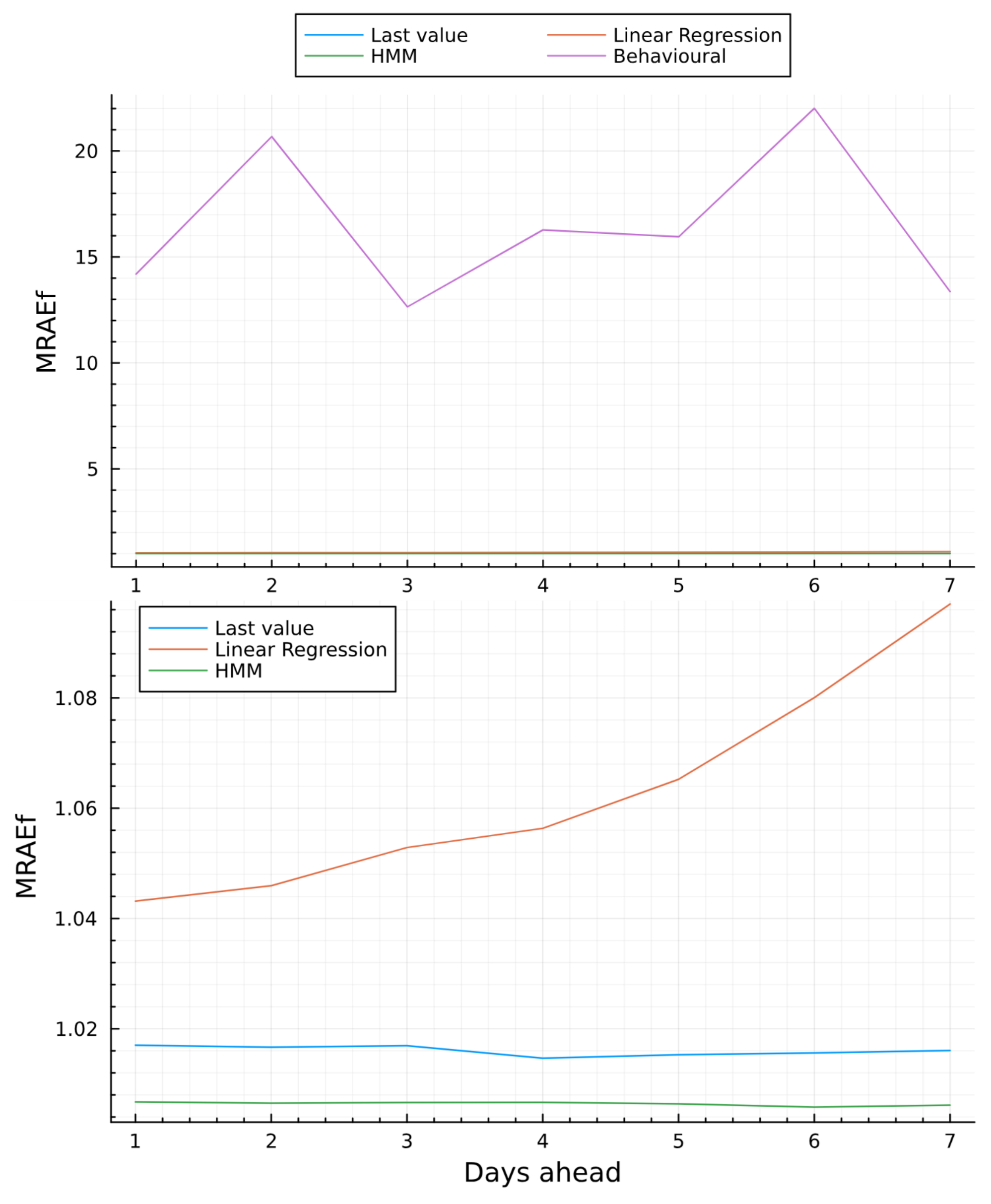
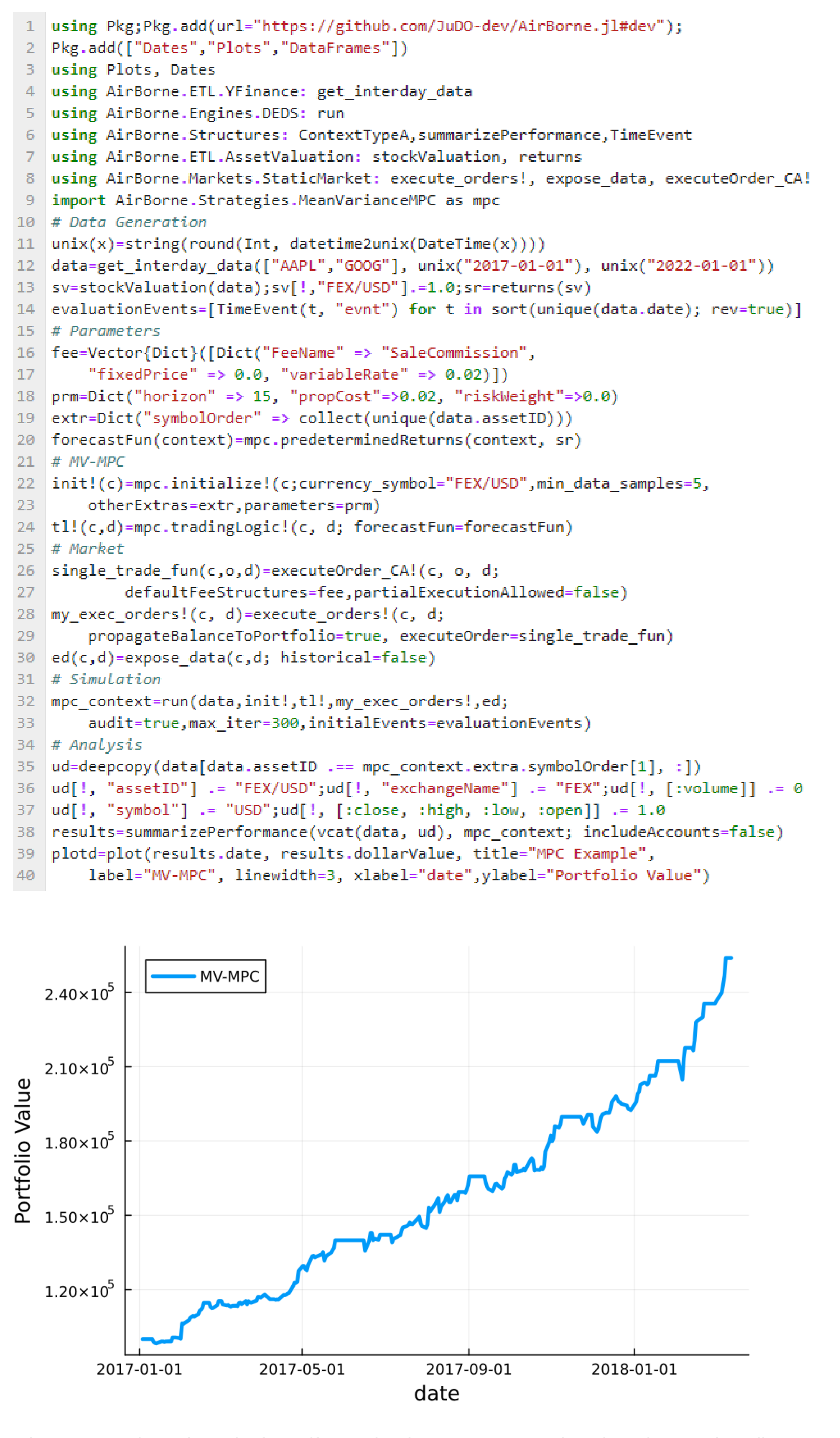


Figure 4: Comparison of Mean Relative Absolute Error (MRAEf) of different forecasting methods predicting the returns of 22 US stocks through 2020 per forecast distance.



```julia
using Pkg;Pkg.add(url="https://github.com/JuDO-dev/AirBorne.jl#dev");
Pkg.add(["Dates","Plots","DataFrames"])
using Plots, Dates
using AirBorne.ETL.YFinance: get_interday_data
using AirBorne.Engines.DEDS: run
using AirBorne.Structures: ContextTypeA,summarizePerformance,TimeEvent
using AirBorne.ETL.AssetValuation: stockValuation, returns
using AirBorne.Markets.StaticMarket: execute_orders!, expose_data, executeOrder_CA!
import AirBorne.Strategies.MeanVarianceMPC as mpc
# Data Generation
unix(x)=string(round(Int, datetime2unix(DateTime(x))))
data=get_interday_data(["AAPL","GOOG"], unix("2017-01-01"), unix("2022-01-01"))
sv=stockValuation(data);sv[!,"FEX/USD"].=1.0;sr=returns(sv)
evaluationEvents=[TimeEvent(t, "evnt") for t in sort(unique(data.date); rev=true)]
# Parameters
fee=Vector{Dict}([Dict("FeeName" => "SaleCommission",
    "fixedPrice" => 0.0, "variableRate" => 0.02)])
prm=Dict("horizon" => 15, "propCost"=>0.02, "riskWeight"=>0.0)
extr=Dict("symbolOrder" => collect(unique(data.assetID)))
forecastFun(context)=mpc.predeterminedReturns(context, sr)
# MV-MPC
init!(c)=mpc.initialize!(c;currency_symbol="FEX/USD",min_data_samples=5,
    otherExtras=extr,parameters=prm)
tl!(c,d)=mpc.tradingLogic!(c, d; forecastFun=forecastFun)
# Market
single_trade_fun(c,o,d)=executeOrder_CA!(c, o, d;
        defaultFeeStructures=fee,partialExecutionAllowed=false)
my_exec_orders!(c, d)=execute_orders!(c, d;
    propagateBalanceToPortfolio=true, executeOrder=single_trade_fun)
ed(c,d)=expose_data(c,d; historical=false)
# Simulation
mpc_context=run(data,init!,tl!,my_exec_orders!,ed;
    audit=true,max_iter=300,initialEvents=evaluationEvents)
# Analysis
ud=deepcopy(data[data.assetID .== mpc_context.extra.symbolOrder[1], :])
ud[!, "assetID"] .= "FEX/USD";ud[!, "exchangeName"] .= "FEX";ud[!, [:volume]] .= 0
ud[!, "symbol"] .= "USD";ud[!, [:close, :high, :low, :open]] .= 1.0
results=summarizePerformance(vcat(data, ud), mpc_context; includeAccounts=false)
plotd=plot(results.date, results.dollarValue, title="MPC Example",
    label="MV-MPC", linewidth=3, xlabel="date",ylabel="Portfolio Value")
```

Figure 5: Code and results for self-contained MV-MPC example using AirBorne in Julia, trading in Google and Apple through 2017 with a 2% transaction fee and perfect information forecast.

[1] X. Li, A. S. Uysal, and J. M. Mulvey, "Multi-period portfolio optimization using model predictive control with mean-variance and risk parity frameworks," European Journal of Operational Research, vol. 299, no. 3, pp. 1158–1176, 2022, issn: 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.10.002
[2] B. Castro, E. Kerrigan and L. Nita and, AirBorne: A Julia Algorithmic Trading Framework, version 0.0.1, Sep. 2023. [Online]. Available: https://github.com/JuDO-dev/AirBorne.jl